**2 - Measuring Distances – Between Observations and Variables**

There are several unsupervised learning methods that start with some measure of distance between observations (*n*) or variables (*p*). In the next two chapters we will be examining multidimensional scaling (a dimension reduction technique) and cluster analysis. Both of these methods require a distance or similarity matrix between observations or variables as the starting point. In this handout we will examine methods for measuring such distances in statistics.

R Packages used in this chapter: proxy, cab, cluster, vegan (library in base R),

**2.1 – Measuring Distance when All Variables are Numeric/Continuous**  
  
To begin, consider the example below which is also included in ***Appendix 1 – Basic Matrix Algebra in R***. This example looks at issues regarding distances between observations using a small set of numeric variables.

**Example 2.1: U.S. Cities**

Measuring Distance/Similarity Between Cities (distance between observations)  
On the basis of these three measured characteristics (median income, percent of population receiving welfare, and percentage of the population below the poverty line) how can we measure how different or similar two cities are, e.g. Detroit, MI and Minneapolis, MN?

The Euclidean distance between two vectors is given by

> View(City)

> X = as.matrix(City[,c(15,17,18)]

> xd = X[9,] 🡨 extract data for Detroit (9th row in the data frame)

> xd

income welfare poverty

18742.0 26.1 32.4

> xm = X[47,] 🡨 extract data for Minneapolis (47th row in the data frame)  
> xm

income welfare poverty

25324.0 10.5 18.5

> t(xd-xm)%\*%(xd-xm) 🡨 Euclidean distance squared

[,1]

[1,] 43323161

> sqrt(t(xd-xm)%\*%(xd-xm)) 🡨 Euclidean distance

[,1]

[1,] 6582.033

Calculations “by hand”:

= 6582.03

Clearly the distance between Detroit and Minneapolis (and any other two cities for that matter) is dominated by the median income. As a result, the discrepancies between the percentages on welfare and below poverty level have little to do with the total dissimilarity between these two cities on the basis of these characteristics.

If we standardize the variables first, we put them all on the same scale.  
> sX = scale(X)

> sxd = sX[9,] 🡨 Detroit

> sxd

income welfare poverty

-1.451120 3.182602 2.410516

> sxm = sX[47,] 🡨 Minneapolis

> sxm

income welfare poverty

-0.29123182 0.05672995 0.08767880

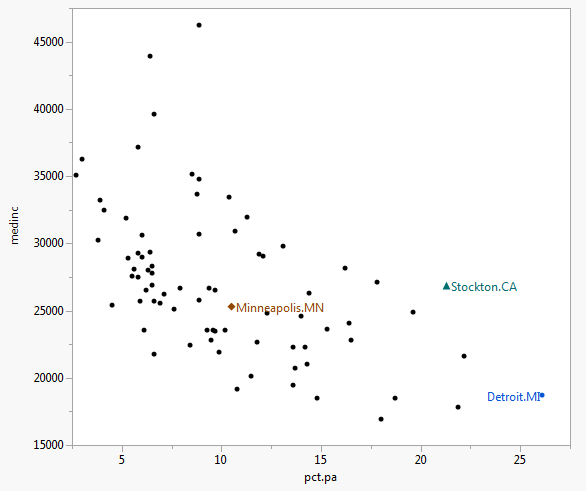
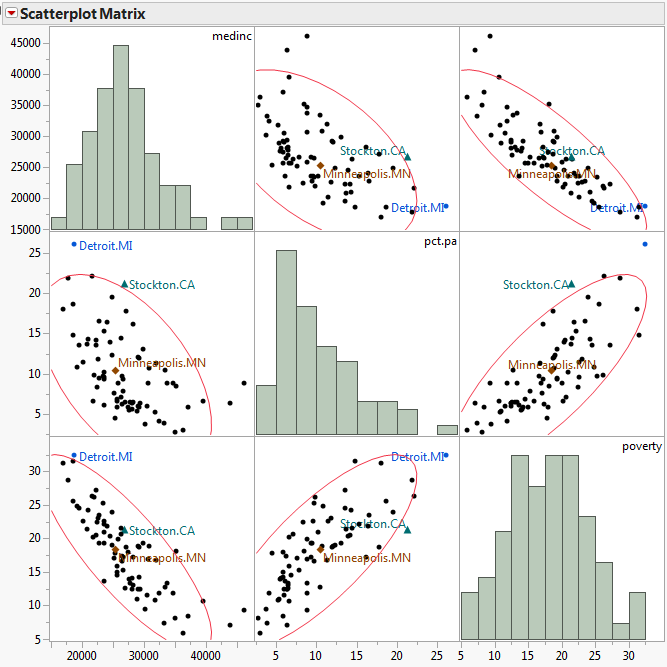
> sqrt(t(sxd-sxm)%\*%(sxd-sxm))

[,1]

[1,] 4.063495

= 4.0635

In the standardized scale the discrepancies between the percentages on welfare and below the poverty level are the largest contributors to the distance between Detroit and Minneapolis. Using a similar process the Euclidean distance between Detroit and Stockton, CA is 8134 and the standardized scale the distance is 2.52. Given the scatterplot below, which measure of discrepancy/distance is more appropriate? Note: These plots were created in JMP.

* *

As this example illustrates, simple Euclidean distance is not appropriate when variables involved in measure distances between observations are on different scales. For this reason standardizing or scaling numeric/quantitative variables is almost done before measuring distances.

Euclidean distance, on scaled or unscaled variables, is not the only way in which we can measure distance between observations based upon a set of *p* exclusively numeric/quantitative variables. Below are some examples of other metrics that are used.

Distances between observations based on numeric variables

**Euclidean Distance**

**Minkowski Distance**

Note: *m = 2* is Euclidean distance

**Manhattan Distance** (Minkowski *m = 1* or *Taxi cab* metric)

**Chebyshev’s Distance**

**Canberra Distance**

What about other variable types, e.g. ordinal or nominal? Can they be used in computing distances between observations and/or variables? Next we will examine how to measure distance using these variable types and then how to calculate dissimilarities based on set variables with a mixture of data types.

**2.2 - Measuring Distance between Observations using Ordinal Variables**

Ordinal variables are often represented as contiguous integers, and the realizable values are considered to be an ordered set , assuming the ordinal variable has levels. To compute distance we replace the values by:

and we can then treat them as numeric/quantitative variables on this transformed scale.

*THIS IS CHEATING…*

*Strongly Disagree = 1, Disagree = 2, Neither = 3, Agree = 4, Strongly Agree = 5*

*To measure, convert to*

*Strongly Disagree = (1-1)/(5-1) = 0, Disagree = (2-1)/(5-1) = 0.25, Neither = (3-1)/(5-1) = 0.5, Agree = (4-1)/(5-1) = 0.75, Strongly Agree = (5-1)/(5-1) = 1*  
 **2.3 - Measuring distance between observations using categorical/nomimal variables**  
  
With unordered categorical variables we could simply use 1 if they agree and 0 otherwise. For example, I believe JMP uses this approach.

If there are several categorical variables involved in measuring the distance between observations, then we can count the number of matches between observations *i* and *j* relative to the “total” number of potential matches. Total is in quotes here because we may want to give more credit for certain type of matches, particularly if all the nominal variables are binary/dichotomous (i.e. have only 2 levels). It is important to note any multilevel nominal/categorical variable can be turned into a sequence of binaries, one for each level. However, doing this will create a great deal of 0-0 matches! Why?

There are several indices in the literature that are used to measure similarity/dissimilarity between observations when we have a set of variables that are all binary or have been turned into binary levels. They all start essentially with a table that counts the number of matches and mismatches.

**Observation *j***

|  |  |  |
| --- | --- | --- |
| **Observation *i*** | 1 (TRUE) | 0 (FALSE) |
| 1 (TRUE) |  |  |
| 0 (FALSE) |  |  |

a = # of variables/levels where observations *i* and *j* are a 1-1 match.  
b = # of variables/levels where observation *i* is 1, but *j* is 0

c = # of variables/levels where observation *i* is 0, but *j* is 1

d = # of variables/levels where observations *i* and *j* are a 0-0 match.  
  
  
On the basis of these counts we can measure ***similarity/dissimilarity*** a number of ways.

Matching Coefficient (dissimilarity)

Jaccard Coefficient (dissimilarity)

this measure completely ignores 0-0 matches

Another way to measure dissimilarity is to convert similarity measures to dissimilarity measures. Below are ways to measure ***similarities*** between observations *i* and *j* based on a set of nominal variables. In the table below (*p = a + b + c + d*).

|  |  |
| --- | --- |
| **Coefficient of Similarity** | **Rationale** |
|  | Equal weights for 1-1 matches and 0-0 matches. |
|  | Double weight for 1-1 matches and 0-0 matches. |
|  | Double weight for unmatched pairs |
|  | No 0-0 matches in numerator |
|  | No 0-0 matches in numerator or denominator.  (The 0-0 matches are treated as irrelevant). |
|  | No 0-0 matches in numerator or denominator.  Double weight for 1-1 matches. |
|  | No 0-0 matches in numerator or denominator.  Double weight for unmatched pairs. |
|  | Ratio of matches to mismatches with 0-0 matches excluded. |

There are at least a dozen more besides these, but I think you probably get the general idea. In many applications the 1-1 matches are far more important than the 0-0 matches and several common measures reflect that fact. The dist()function in the proxy library has 20 different options for measuring similarities (and hence dissimilarities) between observations based on binary variables or binary level indicators for variables with more than two levels.

When a variable is nominal and has *k* levels , we can create *k* binary variables, one for each level of the variable. We can then use the binary measures of similarity above, however it will be very important to discount 0-0 matches as there will be at least of them for a nominal variable with levels.

**Example 2.2: Edibility of Mushrooms and their Physical Characteristics**

**Aside:** These data were used in the 2001 Undergraduate Data Analysis Contest (UDAC) which is now defunct (replaced by MUDAC). The goal of the problem using these data was to develop a predictive model to classify a mushroom as poisonous or edible on the basis of a set of nominal characteristics of the mushrooms. Andrea (Nibbe) Storlie, a WSU alum, earned 1st place in both aspects of the competition (written summary or her results and the accuracy of her predictions – 100% correct!). She went on to become the first undergraduate to win the prestigious Gertrude Cox Scholarship which awarded by the American Statistical Association (ASA) to the best female statistics student in the country.

The mushroom attributes are defined below:

Variable Levels \*  
     Poisonous edible=e,poisonous=p

X1 = cap-shape: bell=b,conical=c,convex=x,flat=f,  
 knobbed=k,sunken=s  
 X2 = cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s  
 X3 = cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r,  
 pink=p,purple=u,red=e,white=w,yellow=y  
 X4 = bruises?: true=t,false=f  
 X5 = odor: almond=a,anise=l,creosote=c,fishy=y,foul=f,  
 musty=m,none=n,pungent=p,spicy=s  
 X6 = gill-attachment: attached=a, free=f  
 X7 = gill-spacing: close=c,crowded=w  
 X8 = gill-size: broad=b,narrow=n  
 X9 = gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g,  
 green=r,orange=o,pink=p,purple=u,red=e,  
 white=w,yellow=y  
 X10 = stalk-shape: enlarging=e,tapering=t  
 X11 = stalk-root: bulbous=b,club=c,cup=u,equal=e,  
 rhizomorphs=z,rooted=r,missing=? 🡨 has lots of missing values!  
 X12 = stalk-surface-above-ring: ibrous=f,scaly=y,silky=k,smooth=s  
 X13 = stalk-surface-below-ring: ibrous=f,scaly=y,silky=k,smooth=s 🡨 has lots of missing values!  
 X14 = stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,  
 pink=p,red=e,white=w,yellow=y  
 X15 = stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,  
 pink=p,red=e,white=w,yellow=y  
 X16 = veil-type: partial=p,universal=u 🡨 All partial (no universal)!  
 X17 = veil-color: brown=n,orange=o,white=w,yellow=y  
 X18 = ring-number: none=n,one=o,two=t  
 X19 = ring-type: cobwebby=c,evanescent=e,flaring=f,large=l,  
 none=n,pendant=p,sheathing=s,zone=z 🡨 some levels not represented  
 X20 = spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r,  
 orange=o,purple=u,white=w,yellow=y  
 X21 = population: abundant=a,clustered=c,numerous=n,  
 scattered=s,several=v,solitary=y  
 X22 = habitat: grasses=g,leaves=l,meadows=m,paths=p,  
 urban=u,waste=w,woods=d

The data frame Mushrooms.train in the mult.Rdata folder I sent you and contains these data for 4062 mushrooms.

In Chapter 3 – Multidimensional Scaling we will conduct a multidimensional scaling of these mushrooms on the basis of most of these variables using similarity/dissimilarity measures for nominal variables. For now however, will consider applying some of the distance measures discussed above applied to these data using two of the physical characteristics of these mushrooms. Let’s consider whether or not the mushroom has bruises ( and the stalk surface above the ring ( and put them in a new data frame called mush.sub.

> table(Mushrooms.train$x4) # 2 levels

f t

2387 1675

> table(Mushrooms.train$x12) # 4 levels

f k s y

286 1176 2585 15

> mush.sub = data.frame(Bruises=Mushrooms.train$x4,SSAR=Mushrooms.train$x12)

> mush.sub[1000:1005,] 🡨 look at the values for mushrooms 1000 - 1005

Bruises SSAR

1000 t s

1001 f k

1002 f s

1003 f k

1004 t s

1005 t s

Use fastDummies library and dummy\_cols() for creating dummy variables in R

As SSAR ( has 4 levels, we can create a dummy variable for each level using the as.dummy command in the cab package. This function will create a dummy for each level of all nominal variables, including dichotomous ones.

> mush.dummy = as.dummy(mush.sub) 🡨 requires cab package installed and loaded.

> mush.dummy[1000:1005,]

Bruises f Bruises t SSAR f SSAR k SSAR s SSAR y

1000 FALSE TRUE FALSE FALSE TRUE FALSE

1001 TRUE FALSE FALSE TRUE FALSE FALSE

1002 TRUE FALSE FALSE FALSE TRUE FALSE

1003 TRUE FALSE FALSE TRUE FALSE FALSE

1004 FALSE TRUE FALSE FALSE TRUE FALSE

1005 FALSE TRUE FALSE FALSE TRUE FALSE

Compute the Jaccard similarity/dissimilarity between mushrooms 1000 and 1001.

Compute the Jaccard similarity/dissimilarity between mushrooms 1002 and 1003.

Compute the Jaccard similarity/dissimilarity between mushrooms 1004 and 1005.

**Relationship Between Similarities and Dissimilarities**   
In general we can turn similarities into dissimilarities intuitively by subtracting them from 1, assuming the similarity between an observation and itself is 1, i.e. max similarity = 1. Thus we have, , although some recommend using or  
 .

We can use R to compute the dissimilarity/similarity matrix for all pairs of mushrooms in our small data set mush.sub using functions in the library proxy. We can display the available similarity/dissimilarity measures (along with their formulae if we want) as shown below.

> library(proxy)  
> summary(pr\_DB,"long")

\* Similarity measures:

Jaccard/binary/Reyssac/Roux (binary) = a / (a + b + c)

Kulczynski1 (binary) = a / (b + c)

Kulczynski2 (binary) = [a / (a + b) + a / (a + c)] / 2

Mountford (binary) = 2a / (ab + ac + 2bc)

Fager/McGowan (binary) = a / sqrt((a + b)(a + c)) - sqrt(a + c) / 2

Russel/Rao (binary) = a / n

simple matching/Sokal/Michener (binary) = (a + d) / n

Hamman (binary) = ([a + d] - [b + c]) / n

Faith (binary) = (a + d/2) / n

Tanimoto/Rogers (binary) = (a + d) / (a + 2b + 2c + d)

Dice/Czekanowski/Sorensen (binary) = 2a / (2a + b + c)

Phi (binary) = (ad - bc) / sqrt[(a + b)(c + d)(a + c)(b + d)]

Stiles (binary) = log(n(|ad-bc| - 0.5n)^2 / [(a + b)(c + d)(a + c)(b + d)])

Michael (binary) = 4(ad - bc) / [(a + d)^2 + (b + c)^2]

Mozley/Margalef (binary) = an / (a + b)(a + c)

Yule (binary) = (ad - bc) / (ad + bc)

Yule2 (binary) = (sqrt(ad) - sqrt(bc)) / (sqrt(ad) + sqrt(bc))

Ochiai (binary) = a / sqrt[(a + b)(a + c)]

Simpson (binary) = a / min{(a + b), (a + c)}

Braun-Blanquet (binary) = a / max{(a + b), (a + c)}

cosine/angular (metric) = xy / sqrt(xx \* yy)

eJaccard/extended\_Jaccard (metric) = xy / (xx + yy - xy)

correlation (metric) = xy / sqrt(xx \* yy) for centered x,y

Chi-squared (nominal) = sum\_ij (o\_i - e\_i)^2 / e\_i

Phi-squared (nominal) = [sum\_ij (o\_i - e\_i)^2 / e\_i] / n

Tschuprow (nominal) = sqrt{[sum\_ij (o\_i - e\_i)^2 / e\_i] / n / sqrt((p - 1)(q - 1))}

Cramer (nominal) = sqrt{[Chi / n)] / min[(p - 1), (q - 1)]}

Pearson/contingency (nominal) = sqrt{Chi / (n + Chi)}

Gower (other) = Sum\_k (s\_ijk \* w\_k) / Sum\_k (d\_ijk \* w\_k)

\* Distance measures:

Euclidean/L2 (metric) = sqrt(sum\_i (x\_i - y\_i)^2))

Mahalanobis (metric) = sqrt((x - y) Sigma^(-1) (x - y))

Bhjattacharyya (metric) = sqrt(sum\_i (sqrt(x\_i) - sqrt(y\_i))^2))

Manhattan/City-Block/L1/taxi (metric) = sum\_i |x\_i - y\_i|

supremum/max/maximum/Tschebyscheff/Chebyshev (metric) = max\_i |x\_i - y\_i|

Minkowski/Lp (metric) = (sum\_i (x\_i - y\_i)^p)^(1/p)

Canberra (metric) = sum\_i |x\_i - y\_i| / |x\_i + y\_i|

Wave/Hedges (metric) = sum\_i (1 - min(x\_i, y\_i) / max(x\_i, y\_i))

divergence (metric) = sum\_i (x\_i - y\_i)^2 / (x\_i + y\_i)^2

Kullback/Leibler (metric) = sum\_i [x\_i \* log((x\_i / sum\_j x\_j) / (y\_i / sum\_j y\_j)) / sum\_j   
 x\_j)]

Bray/Curtis (metric) = sum\_i |x\_i - y\_i| / sum\_i (x\_i + y\_i)

Soergel (metric) = sum\_i |x\_i - y\_i| / sum\_i max{x\_i, y\_i}

Levenshtein (other) = Number of insertions, edits, and deletions between to strings

Podani/discordance (metric) = 1 - 2 \* (a - b + c - d) / (n \* (n - 1))

Chord (metric) = sqrt(2 \* (1 - xy / sqrt(xx \* yy)))

Geodesic (metric) = arccos(xy / sqrt(xx \* yy))

Whittaker (metric) = sum\_i |x\_i / sum\_i x - y\_i / sum\_i y| / 2

Hellinger (metric) = sqrt(sum\_i (sqrt(x\_i / sum\_i x) - sqrt(y\_i / sum\_i y)) ^ 2)

fJaccard/fuzzy\_Jaccard (metric) = sum\_i (min{x\_i, y\_i} / max{x\_i, y\_i})

> summary(pr\_DB,"short")

\* Similarity measures:

Braun-Blanquet, Chi-squared, correlation, cosine, Cramer, Dice, eJaccard, Fager, Faith, Gower,

Hamman, Jaccard, Kulczynski1, Kulczynski2, Michael, Mountford, Mozley, Ochiai, Pearson, Phi,

Phi-squared, Russel, simple matching, Simpson, Stiles, Tanimoto, Tschuprow, Yule, Yule2

\* Distance measures:

Bhjattacharyya, Bray, Canberra, Chord, divergence, Euclidean, fJaccard, Geodesic, Hellinger,

Kullback, Levenshtein, Mahalanobis, Manhattan, Minkowski, Podani, Soergel, supremum, Wave,

Whittaker

Example 2.2: Mushrooms (con’td)  
We will use the Jaccard index as our similarity metric as this metric ignores (False/False or 0-0) matches in both the numerator and the denominator, i.e.

where,

a = # of variables where observations i and j are a 1-1 match.

b = # of variables where observation i is 1, but j is 0

c = # of variables where observation i is 0, but j is 1.

> mush.sim = simil(mush.dummy,method=”Jaccard”) 🡨 these are similarities   
> summary(mush.sim)

Min. 1st Qu. Median Mean 3rd Qu. Max.

0.0000 0.0000 0.3333 0.4363 1.0000 1.0000  
> mush.dist = dist(mush.dummy,method="Jaccard") 🡨 these are dissimilarities

> summary(mush.dist)

Min. 1st Qu. Median Mean 3rd Qu. Max.

0.0000 0.0000 0.6667 0.5637 1.0000 1.0000

If we wish to use the alternative conversions of similarity to dissimilarity ( then we need to convert the similarities to a matrix, perform the conversion, and then express them as a distance matrix.

> mush.sim = simil(mush.dummy,method=”Jaccard”) 🡨 these are similarities   
> summary(mush.sim)

Min. 1st Qu. Median Mean 3rd Qu. Max.

1. 0.0000 0.3333 0.4363 1.0000 1.0000

Convert similarities to a full matrix  
> mush.sim.mat = as.matrix(mush.sim,diag=1)

Convert similarities to distances using the alternative formulae

> mush.dist.mat1 = sqrt(1-mush.sim.mat)

> mush.dist.mat2 = sqrt(2\*(1-mush.sim.mat))

Convert full distance matrices back to lower-triangular storage  
> mush.dist1 = as.dist(mush.dist.mat1)

> mush.dist2 = as.dist(mush.dist.mat2)

Summarize the distances in the alternative scales  
> summary(mush.dist1)

Min. 1st Qu. Median Mean 3rd Qu. Max.

0.0000 0.0000 0.8165 0.6249 1.0000 1.0000

> summary(mush.dist2)

Min. 1st Qu. Median Mean 3rd Qu. Max.

1. 0.0000 1.1550 0.8838 1.4140 1.4140

Compare to original distances   
> summary(mush.dist)

Min. 1st Qu. Median Mean 3rd Qu. Max.

1. 0.0000 0.6667 0.5637 1.0000 1.0000

**2.4 - Combining Distance Measures when Variables are a Mixture of Types**

When the variables in an data matrix are a mixture of types and we wish to measure distance between observations *i* and *j* (i.e. rows *i* and *j* ) we need a method to combine the different distance metrics into an overall distance between them to obtain .

Gower (1971) proposed the following distance metric:

where,

* if either or is missing
* if and variable is asymmetric binary
* otherwise.

The distance portion () is defined as follows according to the data type of variable *k*:

1. If *k* is binary or nominal: if and otherwise.
2. If *k* is numeric: (scaled taxi-cab metric)
3. If *k* is ordinal compute ranks 1,…, and compute

and treating them as numeric using (2) above.

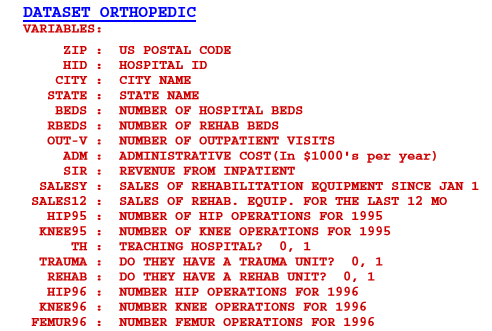
The function daisy()in the cluster library has a Gower’s metric option.

The end result of all of this is an symmetric distance or dissimilarity matrix of the form:

where the diagonal elements are all 0 and the off-diagonal elements are all of the pairwise distances/dissimilarities between each pair of observations . The example of the next page shows an example of Gower’s metric.

**Example of mixture of data types (**uses daisy in the cluster library**)**

**Example 2.3: Orthopedic Equipment Sales**  
One goal in the analysis of these data is to identify hospitals that are underperforming in terms of orthopedic sales given what we know about them. Another thing we might want to look for are groups of similar hospitals on the basis of these characteristics. To do this we need to some measure of similarity/dissimilarity between hospitals given their characteristics. The variables available for this purpose are shown below:



You will notice that three of the variables are binary/dichotomous categorical variables: Teaching Hosptial, Trauma Unit, and Rehabilitation Unit. The others are numeric, and as we will see later, most are very skewed right with lots of zeroes. For the purpose of this example of measuring distance with a mixture of data types we will focus on three dichotomous variables – whether or not the hospital is a teaching hospital, has a trauma unit, and whether or not it has a rehabilitation unit. Because of extreme right skewness with lots zeroes, we will consider the for the numbers of each type of orthopedic surgery conducted at the hospital in the past 12 months.

> Ortho = read.table(file.choose(),header=T,sep=",") # read in **Ortho.csv**  
> names(Ortho)

[1] "Column.1" "V1" "V2" "ZIP" "HospID" "City"

[7] "State" "Beds" "RBeds" "Outpatients" "Admin" "Inpatient"

[13] "Hip95" "Knee95" "SalesYr" "Sales12" "Teach" "Trauma"

[19] "Rehab" "Hip96" "Knee96" "Femur96" "log.Beds." "log.RBeds.1."

[25] "log.Out.1." "log.Admin." "log.Inpat.1." "ln.Hip95.1." "ln.Knee95.1." "ln.Sales.1."

[31] "ln.Sales12.1." "ln.Hip96.1." "ln.Knee96.1." "ln.Fem96.1."

> Ortho.test = Ortho[1:10,c(17:19,32:34)] 🡨 extract info for first 10 hospitals in our data

on the basis of the six variables discussed above.

> Ortho.test

Teach Trauma Rehab ln.Hip96.1. ln.Knee96.1. ln.Fem96.1.

1 0 0 0 2.008600 2.139879 2.004321

2 1 0 0 1.826075 1.612784 2.519828

3 0 0 0 1.991226 1.812913 1.770852

4 0 0 0 2.184691 1.982271 2.068186

5 1 1 0 2.222716 2.045323 2.374748

6 1 1 1 1.763428 1.653213 1.612784

7 0 0 1 0.000000 0.000000 0.000000

8 1 0 0 2.139879 1.924279 2.238046

9 1 0 0 2.318063 2.252853 2.250420

10 1 1 0 1.869232 1.556303 2.161368

> ortho.distance = daisy(Ortho.test,metric="gower",type=list(asymm=1:3))

> ortho.distance

Dissimilarities :

1 2 3 4 5 6 7 8 9

2 0.37932210

3 0.08176069 0.36432804

4 0.05708961 0.37448731 0.09221081

5 0.45626904 0.28413609 0.48853676 0.43321028

6 0.57952849 0.40082229 0.53864814 0.58475355 0.27909545

7 0.90294303 0.90072878 0.84162252 0.91078062 0.96819565 0.68910020

8 0.31127201 0.09636660 0.32474213 0.27812071 0.22874306 0.42180957 0.93309172

9 0.32032816 0.15081784 0.38164868 0.31249057 0.23651818 0.45974726 0.97861695 0.05690654

10 0.47629721 0.23718889 0.46430196 0.47242937 0.09084782 0.21772778 0.89248913 0.26210477 0.30763000

Metric : mixed ; Types = A, A, A, I, I, I

Number of objects : 10

In general, if you have *m* sets of variables of the same type, you can compute a dissimilarity matrix based on each set and then combine them into one matrix using the formula below:

In my opinion what to use as weights is a subjective choice and an area of active research. One simply scheme would be to use weights inversely proportional to the average distance in , i.e. .

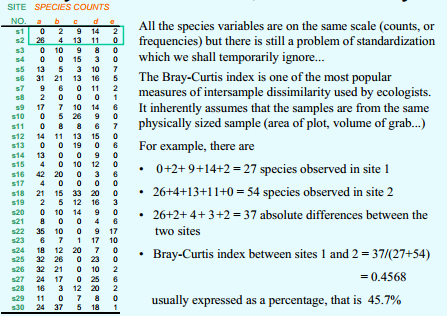
Now that we have a distance matrix representing the dissimilarities, and hence the similarities, between the observations what can we do with it?

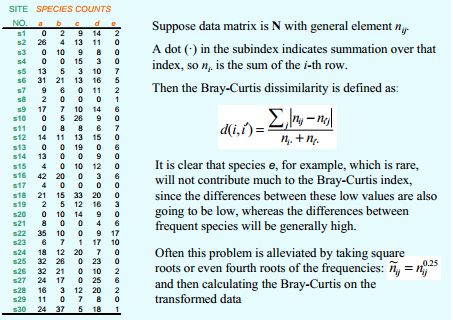
1. We can try to reproduce these distances between observa**t**ions in a lower k-dimensional space (k = 1, 2, or 3 hopefully). By doing so we can identify groups of similar observations (or clusters), look for potential multivariate outliers, and in some cases see what role the variables play in this lower-dimensional orientation or representation of the data. This is the idea behind **multidimensional scaling** or **principal coordinate analysis** (PCoA). To avoid confusion with principal component analysis (PCA) we will refer to this PCoA as metric multidimensional scaling (metric MDS).
2. Perhaps the main use of the pairwise distances between a set of *n* observations is to find groups or *clusters* of similar observations. This is idea behind **cluster analysis** which is sometimes called data segmentation in other disciplines (e.g. market segmentation is finding groups of similar customers or potential customers on the basis of their characteristics).
3. We can also use distances to find the most similar observations in problems where we are trying to estimate something. For example, we might have movie ratings from a large number of individuals and want to predict the rating a person who has not seen a particular movie might give it. We can use distances between this person and other movie watchers in our database to find say five individuals with the most similar rating profiles who have rated the movie in question, and then use their average rating for this movie as our prediction. This is the idea behind nearest neighbor regression and also the idea collaborative filtering/recommender systems which we will discuss later in the course.

**2.4 - Measuring Distance for Observations Based on Counts   
 (**special case of numeric variables**)**

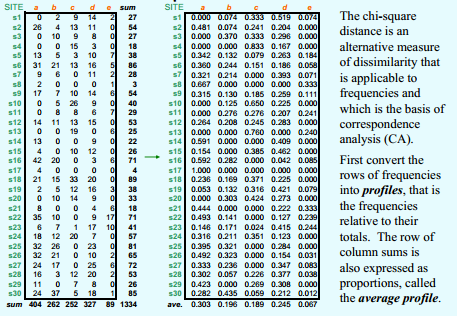
When the set of measurements for an observation represent counts in a specific category, e.g. counts for several species of interest at different sites, then we need to adjust our distance measure relative to total counts for each observation. Two measures that do this are the Bray-Curtis Index of dissimilarity and the Chi-square distance (not to be confused with Mahalanobis’ Distance which is sometimes referred to as *chi-square* distance).

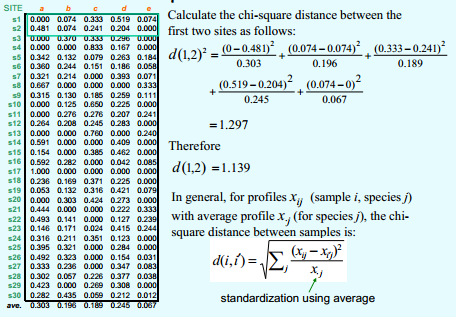
**Bray-Curtis Index**





**Chi-square Distance**





Let proportion counts for observation i in column k, then distance between observations *i* and *j* is given by:

**Example goes here!!!**

**2.5 - Measuring Distance Between Variables**

Above we considered the distance between two observations in a data matrix, i.e. distance between rows. We now turn our attention to the “distance” or “dissimilarity” between two variables, i.e. column vectors in our data matrix. Assuming there are no missing values, two variables have a pair of values for each observation in our sample, and so we can measure the similarity/dissimilarity between these column vectors using a number of potential metrics that are contingent upon the data type of the columns.

**Correlation-based**

One measure of the “similarity” of two variables would be the correlation between them. There are several correlation measures that can be used depending on the data type of variables involved. If both variables *i* and *j* are continuous we could simply use the usual correlation coefficient (i.e. Pearson’s Product Moment Correlation) between them as a measure of similarity.

This measures the degree of linear association between two numeric variables. Alternatively, we could use Spearman’s Rho () which is simply the Pearson’s Product Moment Correlation between and after they have been **rank ordered**. This will dampen the effect of outliers.

As the correlation is a measure of similarity we can convert it to a dissimilarity using the formula we have seen previously,

where here is one of the correlation measures discussed above.

**Distances between count variables**

We can first transpose the data matrix so the rows correspond to variables and the columns correspond to observations and then apply the Bray-Curtis or Chi-square distances discussed above for the measuring distances between observations based on count variables.

**Distances between categorical variables**When both variables are categorical/nominal we can cross-tabulate them (i.e. set up a contingency table) and then use one of the plethora of measures of association between variables summarized in this matter. For example, one measure is that works well is Cramer’s V coefficient.

This is a measure of “similarity” so we can easily convert this is to distance measure by subtracting from 1 or use .

**Measuring distance between variables with mixed types**

Measuring distances between variables that are mixture of data types is not discussed much in the literature, e.g. a continuous variable and a categorical/nominal variable.

There are numerous association/”correlation” measures in the literature for data of mixed types (i.e. similarities) and therefore we could turn those into distance measures using the methods similar to those discussed above.

**2.6 – Discussion**

The concepts of distance and similarity are important as we begin to look at unsupervised learning methods for extracting information from data. In dimension reduction methods, such a multidimensional scaling and principal components, we attempt to take -dimensional data and find a lower-dimensional representation (ideally 1, 2, or 3 dimensions) where distances between observations in -dimensional space are “preserved” in the lower dimensional representation. In cluster analysis we attempt to form homogenous groups of observations/variables based upon distances between them. Observations within the same final group/cluster should be fairly similar to one another. In developing recommender systems we make recommendations to a given user based upon what similar users have liked. As you can see the ideas of dissimilarity and similarity play a central role in several methods we will be exploring in subsequent chapters.